Electronic Communications of the EASST
Volume 18 (2009)

Proceedings of the Eighth International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2009)

Completeness and Correctness of Model Transformations based on Triple Graph Grammars with Negative Application Conditions

Hartmut Ehrig , Frank Hermann and Christoph Sartorius

18 pages

Guest Editors: Artur Boronat, Reiko Heckel
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Completeness and Correctness of Model Transformations based on Triple Graph Grammars with Negative Application Conditions

Hartmut Ehrig[1] **, Frank Hermann**[1] **and Christoph Sartorius**[1]

[1] [ehrig, frank, csart](at)cs.tu-berlin.de
Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin, Germany

Abstract: Model transformations are a key concept for modular and distributed model driven development. In this context, triple graph grammars have been investigated and applied to several case studies and they show a convenient combination of formal and intuitive specification abilities. Especially the automatic derivation of forward and backward transformations out of just one specified set of rules for the integrated model simplifies the specification and enhances usability as well as consistency.

Since negative application conditions (NACs) are key ingredient for many model transformations based on graph transformation we embed them in the concept of triple graph grammars. As a first main result we can extend the composition/decomposition result for triple graph grammars to the case with NACs. This allows us to show completeness and correctness of model transformations based on rules with NACs and furthermore, we can extend the characterization of information preserving model transformations to the case with NACs.

The presented results are applicable to several model transformations and in particular to the well known model transformation from class diagrams to relational data bases, which we present as running example with NACs.

Keywords: model transformation, triple graph grammars, completeness, correctness, negative application conditions

1 Introduction

Model transformations based on triple graph grammars have been introduced in [Sch94, KS06]. In order to define a general framework independent of the specific domain and target language the correspondences between source and target models are defined as relational mappings, where forward and backward transformation rules are derived automatically.

In [EEE+07] we showed how to analyze bi-directional model transformations based on triple graph grammars with respect to information preservation, which is based on a decomposition and composition result for triple graph grammar sequences. Moreover, completeness and correctness of model transformations have been studied on this basis in [EEH08b, EEH08c]. All formal results in these papers, however, do not consider negative application conditions (NACs), which are very important for several practical applications (see [SK08]). The main purpose of this paper is to extend TGGs with NACs on a formal basis.

As a main result we show completeness, correctness and information preservation of model transformations with NACs. Our new result can be used to check, whether a model transformation performed by an algorithm using triple graph transformations with NACs such as [SK08] is correct (see Section 7). The relationship between forward and backward model transformation sequences was analyzed already in [EEE+07] based on a canonical decomposition and composition result for triple transformations and this paper extends it to the case with NACs.

In Section 2 we review triple graphs and introduce the case study for a model transformation from class models to relational data base models. Section 3 reviews triple rules and triple graph transformations as introduced in [Sch94] and extends them to the case with NACs showing that the composition and decomposition result is also valid for this extension. The second main result of correctness and completeness of model transformations based on source consistent model transformations with NACs is presented in Section 4 and explained on a concrete model transformation sequence of the example. Section 5 shows how the characterization of information preserving bidirectional model transformations is extended to the case with NACs. Related and future work are discussed in sections 6 and 7, respectively.

2 Review of Triple Graphs

Triple graph grammars [Sch94] are a well known approach for bidirectional model transformations. Models are defined as pairs of source and target graphs which are connected via an intermediate correspondence graph together with its embeddings into these graphs. In [KS06], Königs and Schürr formalize the basic concepts of triple graph grammars in a set-theoretical way, which was generalized and extended by Ehrig et. el. in [EEE+07] to typed, attributed graphs. In this section, we shortly review triple graphs, while triple rules are defined in Sec. 3 together with the extension to negative application conditions (NACs).

Definition 1 (Triple Graph and Triple Graph Morphism) Three graphs G_S, G_C, and G_T, called source, connection, and target graphs, together with two graph morphisms $s_G : G_C \to G_S$ and $t_G : G_C \to G_T$ form a triple graph $G = (G_S \xleftarrow{s_G} G_C \xrightarrow{t_G} G_T)$. G is called *empty*, if G_S, G_C, and G_T are empty graphs.

A triple graph morphism $m = (s,c,t) : G \to H$ between two triple graphs $G = (G_S \xleftarrow{s_G} G_C \xrightarrow{t_G} G_T)$ and $H = (H_S \xleftarrow{s_H} H_C \xrightarrow{t_H} H_T)$ consists of three graph morphisms $s : G_S \to H_S$, $c : G_C \to H_C$ and $t : G_T \to H_T$ such that $s \circ s_G = s_H \circ c$ and $t \circ t_G = t_H \circ c$. It is injective, if morphisms s, c and t are injective. A typed triple graph G is typed over a triple graph $TG = (TG_S \leftarrow TG_C \to TG_T)$ by a triple graph morphism $t_G : G \to TG$.

Example 1 *The type graph of the example is given in Fig. 1 showing the structure of class diagrams in the source component and relational databases in the target component. Classes correspond to tables and attributes to columns. Throughout the example, originating from [SK08] and [EEE+07], elements are arranged left, center, and right according to the component*

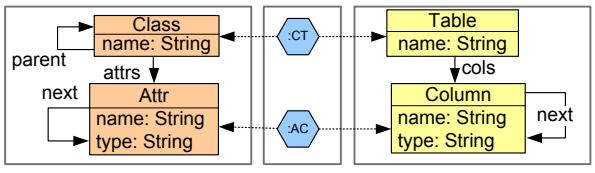


Figure 1: Triple type graph for CD2RDBM

types source, correspondence and target. Morphisms starting at a connection part are given by dotted arrow lines. Note that the case study is equipped with attribution, which is based on the concept of E-graphs [EEPT06].

The extension of the results of this paper to the case with attributes shall be straight forward, all results can be shown in the framework of weak adhesive HLR categories and hence, also for the category $\mathbf{AGraphs}_{ATG}$ of attributed graphs.

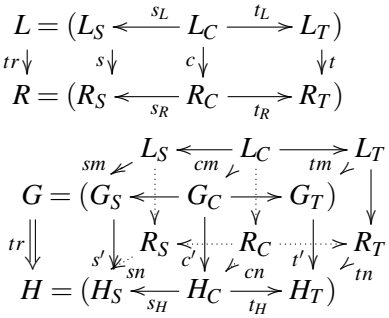
3 Triple Graph Grammars with NACs

Many model transformations use the concept of negative application conditions (NACs) introduced in [HHT96]. NACs can ensure termination and they can control the application of model transformation rules by defining forbidden structures as extensions of left hand sides of rules. If a forbidden structure is present around the selected match, the corresponding rule is not applicable and the match is invalid, i.e. NACs restrict the applicability of model transformation rules.

While triple graph grammars (TGGS) are an elegant way to descriptively define model transformations by defining triple rules that specify the synchronous creation of source and target model, formal results are mainly given for the case of TGGs without NACs. In this section we review triple rules, derivation of transformation rules and we define NACs for triple rules. The case study presents rules with NACs motivated by a similar model transformation in [SK08], where NACs are used to ensure well formed list structures.

A triple rule is used to build up source and target graphs as well as their connection graph, i.e. they are non-deleting. Structure filtering which deletes parts of triple graphs, is performed by projection operations only, i.e. structure deletion is not done by rule applications.

Definition 2 (Triple Rule tr and Triple Transformation Step) A *triple rule* tr consists of triple graphs L and R, called left-hand and right-hand sides, and an injective triple graph morphism $tr = (s,c,t) : L \to R$. Given a triple rule $tr = (s,c,t) : L \to R$, a triple graph G and an injective triple graph morphism $m = (sm,cm,tm) : L \to G$, called triple match m, a triple graph transformation step (TGT-step) $G \overset{tr,m}{\Longrightarrow} H$ from G to a triple graph H is given by three pushouts (H_S,s',sn), (H_C,c',cn) and (H_T,t',tn) in category **Graph** with induced morphisms s_H : $H_C \to H_S$ and $t_H : H_C \to H_T$. Morphism $n = (sn,cn,tn)$ is called comatch.



Moreover, we obtain a triple graph morphism $d : G \to H$ with $d = (s',c',t')$ called transformation morphism. A sequence of triple graph transformation steps is called triple (graph) transformation sequence, short: TGT-sequence. Furthermore, a triple graph grammar $TGG = (S,TR)$ consists of a triple start graph S and a set TR of triple rules. Given a triple rule tr we refer by $L(tr)$ to its left and by $R(tr)$ to its right hand side.

Definition 3 (Triple, Source and Target Language) A set of triple rules TR defines the *triple language* $VL = \{G \mid \emptyset \Rightarrow^* G \text{ via } TR\}$ of triple graphs. *Source language* VL_S and *target language* are derived by projection to the triple components, i.e. $VL_S = proj_S(VL)$ and $VL_T = proj_T(VL)$,

where $proj_X$ is a projection defined by restriction to one of the triple components, i.e. $X \in \{S,C,T\}$.

Definition 4 (Derived Triple Rules) From each triple rule $tr = L \to R$ we have the following source, forward, target and backward rules:

$$\begin{array}{ccc} L_S & \leftarrow \emptyset \rightarrow & \emptyset \\ {\scriptstyle s}\downarrow & \downarrow & \downarrow \\ R_S & \leftarrow \emptyset \rightarrow & \emptyset \end{array} \qquad \begin{array}{ccc} \emptyset & \leftarrow \emptyset \rightarrow & L_T \\ \downarrow & \downarrow & {\scriptstyle t}\downarrow \\ \emptyset & \leftarrow \emptyset \rightarrow & R_T \end{array} \qquad \begin{array}{ccc} R_S & \xleftarrow{s\circ s_L} & L_C \xrightarrow{t_L} L_T \\ {\scriptstyle id}\downarrow & {\scriptstyle c}\downarrow & \downarrow{\scriptstyle t} \\ R_S & \xleftarrow{s_R} & R_C \xrightarrow{t_R} R_T \end{array} \qquad \begin{array}{ccc} L_S & \xleftarrow{s_L} & L_C \xrightarrow{t\circ t_L} R_T \\ {\scriptstyle s}\downarrow & {\scriptstyle c}\downarrow & {\scriptstyle id}\downarrow \\ R_S & \xleftarrow{s_R} & R_C \xrightarrow{t_R} R_T \end{array}$$

source rule tr_S | target rule tr_T | forward rule tr_F | backward rule tr_B

Source rules allow to create all elements of VL_S as restriction of VL, but they contain less restrictions for matches during transformation in comparison to their corresponding complete triple rules. Thus, they possibly allow to generate more elements than VL_S contains. This means that in general we have inclusion $VL_S \subseteq VL_{S0} = \{G_S \,|\, \emptyset \Rightarrow^* G_S \; via \; TR_S\}$ resp. $VL_T \subseteq VL_{T0} = \{G_T \,|\, \emptyset \Rightarrow^* G_T \; via \; TR_T\}$, where TR_S and TR_T are the sets of source resp. target rules derived from TR.

Definition 5 (General Negative Application Condition) Given a triple rule $tr = (L \xrightarrow{tr} R)$, a general negative application condition (NAC) (N,n) consists of a triple graph N and an injective triple graph morphism $n : L \to N$.
A match $m : L \to G$ is NAC consistent if there is no injective $q : N \to G$ such that $q \circ n = m$. A triple transformation $G \overset{*}{\Rightarrow} H$ is NAC consistent if all matches are NAC consistent.

Definition 6 (Source-Target Negative Application Condition) A source-target NAC (N,n) is a NAC with injective triple graph morphism $n : L \to N$ with $n = (n_S, id_{L_C}, id_{L_T})$ or $n = (id_{L_S}, id_{L_C}, n_T)$.
This means a source-target NAC is a NAC which only prohibits the existence of certain structures either in the source (*source NAC*) or in the target part (*target NAC*).

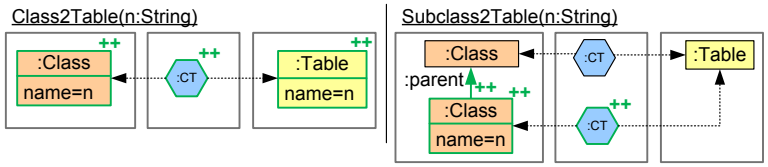


Figure 2: Rules for transforming classes to tables

In most usecases we encounter only source-target NACs, therefore we regard them as the standard case. In the following when speaking of NACs we always mean source-target NACs. If this is not the case we will explicitly refer to the term general NAC.

Definition 7 (Derived Triple Rules with NACs) Given a triple rule tr with NACs and let $\underline{tr}$ be its underlying triple rule without NACs. Let $\underline{tr_S}$, $\underline{tr_T}$, $\underline{tr_F}$ and $\underline{tr_B}$ be the derived rules from $\underline{tr}$ according to Def. 4. Then, source rule tr_S, target rule tr_T, forward rule tr_F and backward rule tr_B are given by the underlying rules $\underline{tr_S}$, $\underline{tr_T}$, $\underline{tr_F}$ and $\underline{tr_B}$, where additionally tr_S as well as tr_B contain all source NACs of tr and tr_T as well as tr_F contain all target NACs of tr.

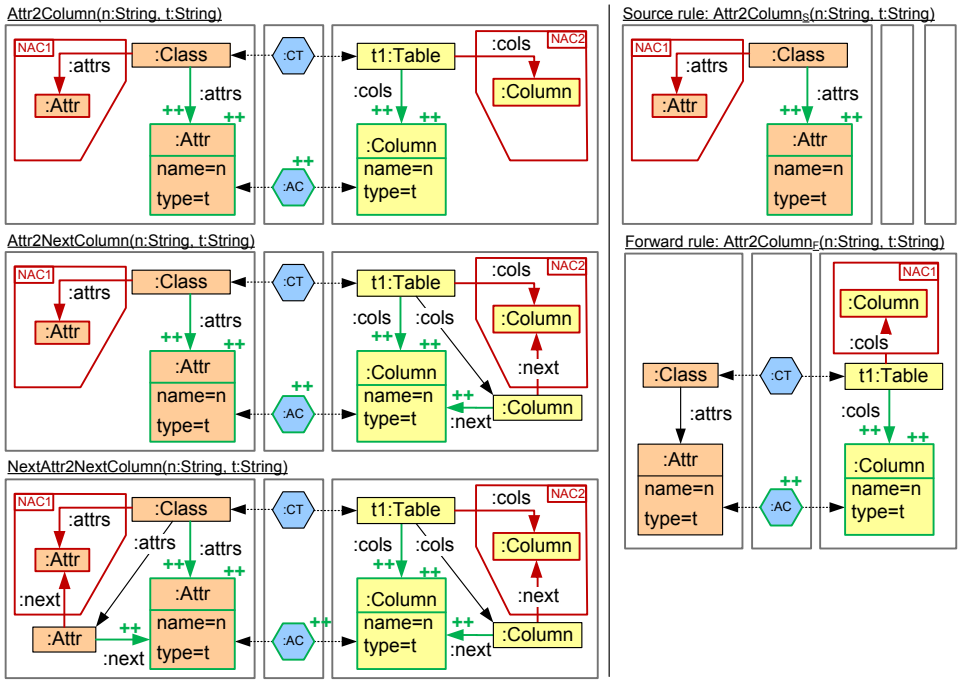


Figure 3: Rules for transforming attributes to columns and derived source and forward rule

Example 2 (Triple Rules) *Examples for triple rules are given in Fig. 2 and Fig. 3 in short notation. Left and right hand side of a rule are depicted in one triple graph. Elements, which are created by the rule, are labeled with green "++" and marked by green line coloring. Rule "*Class2Table*" synchronously creates a class in a class diagram with its corresponding table in the relational database. Accordingly the other rules create parts in all components. NACs are indicated by red frames with label "NAC" around the extension of the left hand side of a rule. Each forward rule is derived from a triple tr rule as follows: The source components which are created in tr are preserved by tr_F, i.e. they are in the left hand side. The source NAC is omitted and the rest of tr keeps the same. For example the forward rule of "Attr2Colum" is derived by omitting "NAC1" and adding to the left hand side the attribute node with its connecting edge to the class node shown on the right part of Fig. 3.*

Theorem 1 as a main technical result of the paper shows that TGT-sequences can be decomposed to source and forward sequences and composed out of them. All together this correspondence is bijective. The result uses the following notion of match consistency.

Definition 8 (Match and Source Consistency) Let tr_S^* and tr_F^* be sequences of source rules tri_S and forward rules tri_F, which are derived from the same triple rules tri for $i = 1,\ldots,n$. Let further $G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n0} \overset{tr_F^*}{\Longrightarrow} G_{nn}$ be a TGT-sequence with (mi_S, ni_S) being match and comatch of tri_S (respectively (mi_F, ni_F) for tri_F) then match consistency of $G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n0} \overset{tr_F^*}{\Longrightarrow} G_{nn}$ means that the S-component of the match mi is uniquely determined by the comatch ni_S $(i = 1,\ldots,n)$.

A TGT-sequence $G_{n0} \overset{tr_F^*}{\Longrightarrow} G_{nn}$ is source consistent, if there is a match consistent sequence $\emptyset \overset{tr_S^*}{\Longrightarrow} G_{n0} \overset{tr_F^*}{\Longrightarrow} G_{nn}$. Note that by source consistency the application of the forward rules is controlled by the source sequence, which generates the given source model.

Theorem 1 (Decomposition and Composition of TGT-Sequences with NACs)

1. **Decomposition**: *For each TGT-sequence*
$$G_0 \overset{tr_1}{\Longrightarrow} G_1 \Rightarrow \ldots \overset{tr_n}{\Longrightarrow} G_n \tag{1}$$
with NACs there is a corresponding match consistent TGT-sequence
$$G_0 = G_{00} \overset{tr_{1S}}{\Longrightarrow} G_{10} \Rightarrow \ldots \overset{tr_{nS}}{\Longrightarrow} G_{n0} \overset{tr_{1F}}{\Longrightarrow} G_{n1} \Rightarrow \ldots \overset{tr_{nF}}{\Longrightarrow} G_{nn} = G_n \tag{2}$$
with NACs.

2. **Composition:** *For each match consistent transformation sequence* (2) *with NACs there is a canonical transformation sequence* (1) *with NACs.*

3. **Bijective Correspondence:** *Composition and decomposition are inverse to each other.*

Remark 1 (Injective matches) *Opposed to the version without NACs in [EEE+07] the matches of the triple rules are required to be injective. If we allow non-injective matches, then we must allow n and q in definition 5 to be non-injective as well.*

In order to prove Thm. 1 we we first show the following Lemma 1 and Lemma 2 thereafter. The first lemma shows the correspondence between valid matches of triple rules and valid matches of their derived forward and source rules.

Lemma 1 *The injective match of a triple rule tr is NAC-consistent if and only if the injective matches of the derived rules tr_S and tr_F are NAC-consistent.*

Proof of Lemma 1. From [EEE+07] we know that any tr is equal to the E-concurrent rule $tr_S \star_E tr_F$ with $E = L_F$.

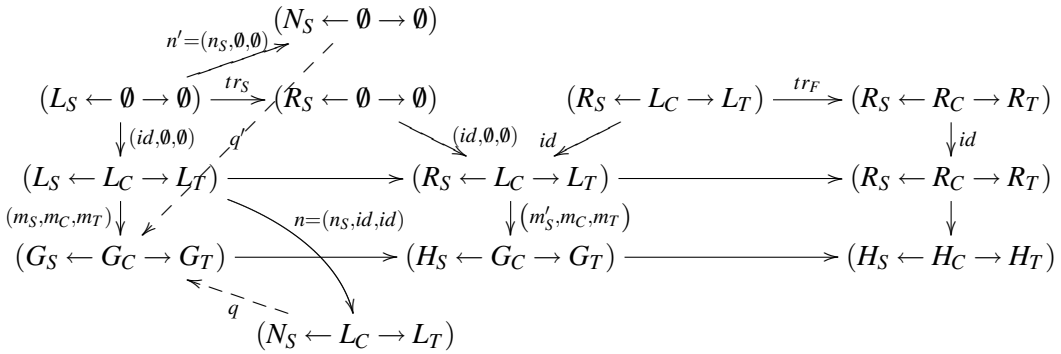


Figure 4: E-concurrent rule with source NAC

Now we consider the NACs of tr and the corresponding NACs of tr_S and tr_F (as described in Def. 7) using this construction (see Fig. 4 resp. Fig. 7). It remains to show that the matches of

the source rule tr_S and the forward rule tr_F are NAC consistent if and only if the match of the triple rule tr is NAC consistent.

- *match of tr is NAC-consistent for source NAC $\Rightarrow$ match of tr_S is NAC-consistent:* Assume tr_S is not NAC consistent $\Rightarrow$ $\exists$ injective $q' : N' \to G$ with $q' = (q'_S, q'_C, q'_T)$ such that $q' \circ n' = (m_S, \emptyset, \emptyset)$. Then we are able to construct an injective morphism $q : N \to G$ with $q = (q'_S, m_C, m_T)$ such that $q \circ n = (m_S, m_C, m_T)$ (Fig. 4). q is a valid triple graph morphism if (1) and (2) commute in Fig. 5. (2) commutes because (m_S, m_C, m_T) is a valid morphism by construction and therefore commutes. $s_G \circ m_C = m_S \circ s_L = q'_S \circ n_S \circ s_L = q'_S \circ s_N \circ id = q'_S \circ s_N$, hence (1) commutes too. This means tr is not NAC consistent $\Rightarrow$ contradiction!

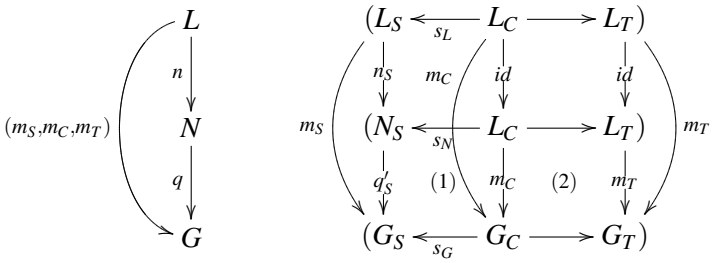


Figure 5: constructed morphism q is valid

- *match of tr_S is NAC-consistent $\Rightarrow$ match of tr is NAC-consistent for source NAC:* Assume tr is not NAC consistent $\Rightarrow$ $\exists$ injective $q : N \to G$ with $q = (q_S, q_C, q_T)$ such that $q \circ n = (m_S, m_C, m_T)$. Then we are able to construct an injective morphism $q' : N' \to G$ with $q' = (q_S, \emptyset, \emptyset)$ such that $q' \circ n' = (m_S, \emptyset, \emptyset)$ (Fig. 4). q' is a valid triple graph morphism if (1) and (2) commute in Fig. 6, which they obviously do. This means tr_S is not NAC consistent $\Rightarrow$ contradiction!

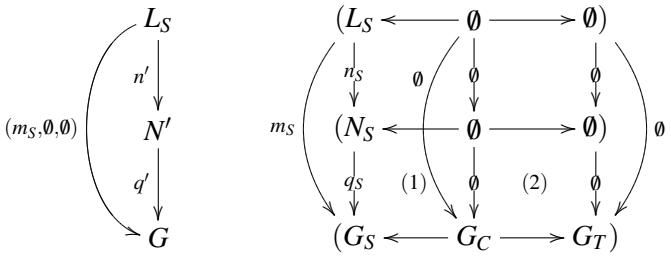


Figure 6: constructed morphism q' is valid

- *match of tr is NAC-consistent for target NAC $\Rightarrow$ match of tr_F is NAC-consistent:* Assume tr_F is not NAC consistent $\Rightarrow$ $\exists$ injective $q' : N' \to G'$ with $q' = (q'_S, q'_C, q'_T)$ such that $q' \circ n' = (m'_S, m_C, m_T)$. Then we are able to construct an injective morphism $q : N \to G$ with $q = (m_S, m_C, q'_T)$ such that $q \circ n = (m_S, m_C, m_T)$ (Fig. 7). q is a valid triple graph

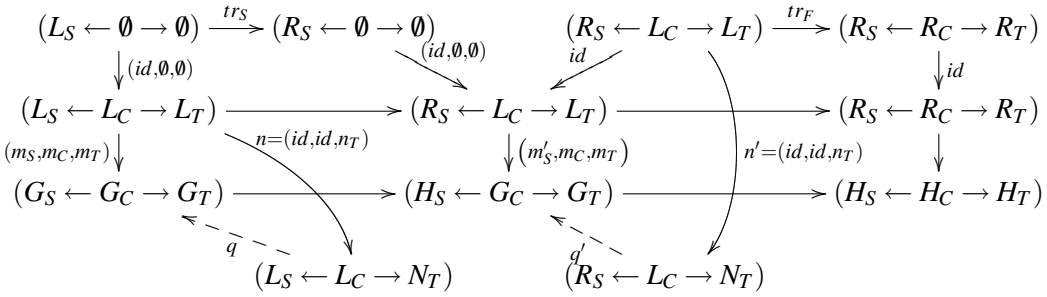


Figure 7: E-concurrent rule with target NAC

morphism if (1) and (2) commute in Fig. 8. (1) commutes because (m_S,m_C,m_T) is a valid morphism by construction and therefore commutes. $t_G \circ m_C = m_T \circ t_L = q_T \circ n_T \circ t_L = q_T \circ t_N \circ id = q_T \circ t_N$, hence (2) commutes too. This means tr is not NAC consistent $\Rightarrow$ contradiction!

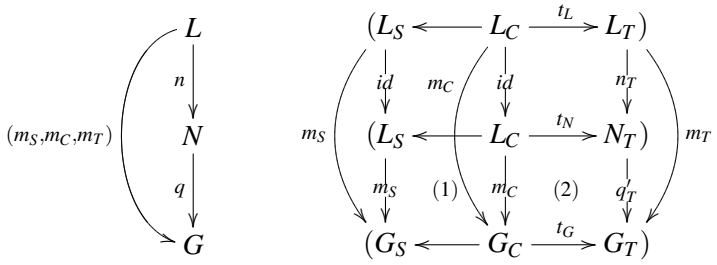


Figure 8: constructed morphism q is valid

- *match of tr_F is NAC-consistent $\Rightarrow$ match of tr is NAC-consistent for target NAC:* Assume tr is not NAC consistent $\Rightarrow$ $\exists$ injective $q : N \rightarrow G$ with $q = (q_S,q_C,q_T)$ such that $q \circ n = (m_S,m_C,m_T)$. Then we are able to construct an injective morphism $q' : N' \rightarrow G'$ with $q' = (m'_S,m_C,q_T)$ such that $q' \circ n' = (m'_S,m_C,m_T)$ (Fig. 7). q' is a valid triple graph morphism if (1) and (2) commute in Fig. 9, which they do analoguously. This means tr_F is not NAC consistent $\Rightarrow$ contradiction!

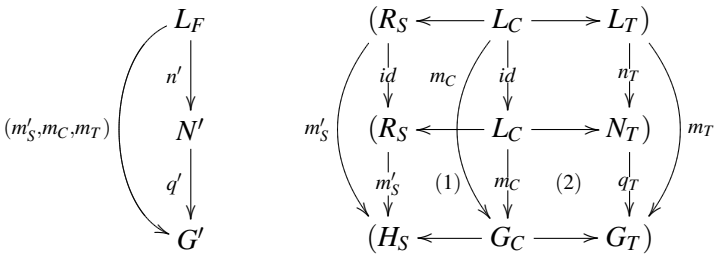


Figure 9: constructed morphism q' is valid

□

The next lemma shows that independent derivation steps of source and forward rules can be switched.

Lemma 2 *Given sequentially independent derivation steps via rules tr_{2S} and tr_{1F} with NACs, then the following holds: The injective matches of $G_{10} \overset{(tr_{1F},m_1)}{\Longrightarrow} G_{11} \overset{(tr_{2S},m_2)}{\Longrightarrow} G_{21}$ are NAC consistent if and only if the injective matches of $G_{10} \overset{(tr_{2S},m_{2'})}{\Longrightarrow} G_{20} \overset{(tr_{1F},m_{1'})}{\Longrightarrow} G_{21}$ are NAC consistent, too.*

Proof of Lemma 2. Having constructed a NAC consistent match consistent sequence (3) we now want to reorder the rules according to Fig. 10 until we have sequence (2). We have to show that upon swapping the rules the NAC consistency is preserved (see Fig. 11 resp. Fig. 13).

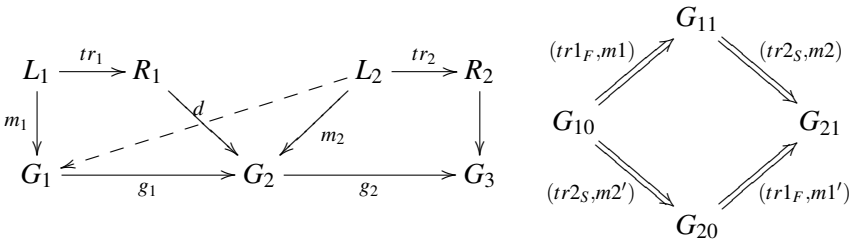


Figure 10: sequential independence of source and forward rules

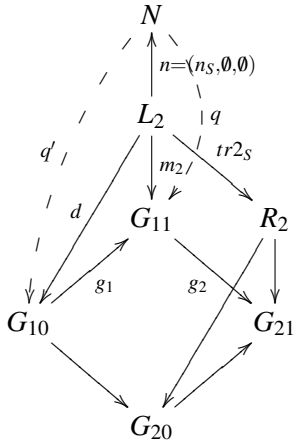


Figure 11: sequential independence: source rule with NAC

- *match m_2 of $tr2_S$ is NAC consistent $\Rightarrow$ match $m_2' = d$ of $tr2_S$ is NAC consistent:* Assume d is not NAC consistent $\Rightarrow$ $\exists$ injective $q' : N \to G_{10}$ with $q' = (q'_S, q'_C, q'_T) = (q'_S, \emptyset, \emptyset)$ such that $q' \circ n = d$. Furthermore we know that $g_1 \circ d = m_2$. Thus $g_1 \circ q' \circ n = m_2$. Because g_1 is based on a forward rule we know that $(G_{10})_S = (G_{11})_S$ and $g_{1_S} = id_{G_{10S}} \Rightarrow$ $\exists$ injective

$q : N \to G_{11}$ with $q = g_1 \circ q' = (q'_S, \emptyset, \emptyset)$ such that $q \circ n = m_2$ because $q \circ n = g_1 \circ q' \circ n = m_2$. q is a valid triple graph morphism if (1) and (2) commute in Fig. 12, which they obviously do. This means m_2 is not NAC consistent $\Rightarrow$ contradiction!

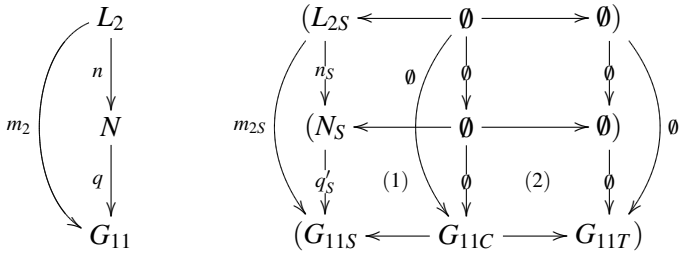


Figure 12: constructed morphism q is valid

- *match d of $tr2_S$ is NAC consistent $\Rightarrow$ match m_2 of $tr2_S$ is NAC consistent:* Assume m_2 is not NAC consistent $\Rightarrow$ $\exists$ injective $q : N \to G_{11}$ with $q = (q_S, q_C, q_T) = (q_S, \emptyset, \emptyset)$ such that $q \circ n = m_2$. We know that $g_1 \circ d = m_2$. Thus $g_1 \circ d = q \circ n$. Because g_1 is based on a forward rule we know that $(G_{10})_S = (G_{11})_S$ and $g_{1_S} = id_{G_{10S}} \Rightarrow d_S = q_S \circ n_S \Rightarrow$ $\exists$ injective $q' : N \to G_{10}$ with $q' = (q_S, \emptyset, \emptyset)$ such that $q' \circ n = d$ because $q'_S \circ n_S = d_S$. q' is a valid triple graph morphism by the same arguments as in Fig. 12. This means d is not NAC consistent $\Rightarrow$ contradiction!

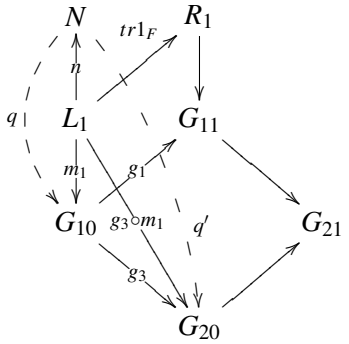


Figure 13: sequential independence: forward rule with NAC

- *match m_1 of $tr1_F$ is NAC consistent $\Rightarrow$ match $m'_1 = g_3 \circ m_1$ of $tr1_F$ is NAC consistent:* Assume m'_1 in Fig. 13 is not NAC consistent $\Rightarrow$ $\exists$ injective $q' : N \to G_{20}$ with $q' = (q'_S, q'_C, q'_T)$ such that $q' \circ n = m'_1$. We know that $(L_1)_S = N_S$ and $(L_1)_C = N_C$ with $n_S = id_{L_{1S}}$ and $n_C = id_{L_{1C}}$. Furthermore g_3 is based on a source rule which means that $(G_{10})_T = (G_{20})_T$ and $g_{3_T} = id_{G_{10T}}$. Thus $\exists$ injective $q : N \to G_{10}$ with $q = (m_{1_S}, m_{1_C}, q'_T)$ such that $q \circ n = m_1$. q is a valid triple graph morphism if (1) and (2) commute in Fig. 14. (1) commutes obviously. $t_{G_{10}} \circ m_{1C} = m_{1T} \circ t_{L_1} = q'_T \circ n_T \circ t_{L_1} = q'_T \circ t_N \circ id = q'_T \circ t_N$, hence (2) commutes too. This means m_1 is not NAC consistent $\Rightarrow$ contradiction!

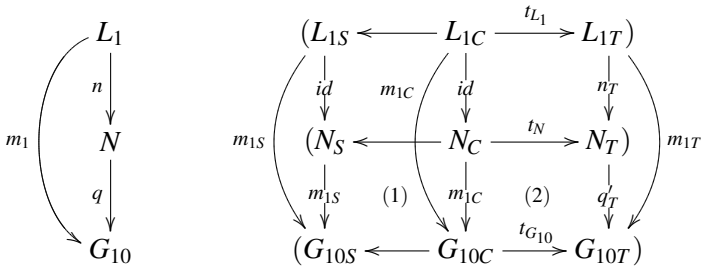
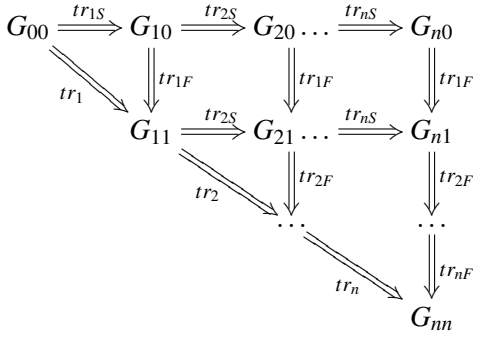


Figure 14: constructed morphism q is valid

- *match $m_1' = g_3 \circ m_1$ of $tr1_F$ is NAC consistent $\Rightarrow$ match m_1 of $tr1_F$ is NAC consistent:* Assume m_1 is not NAC consistent $\Rightarrow$ $\exists$ injective $q : N \to G_{10}$ with $q = (q_S, q_C, q_T)$ such that $q \circ n = m_1$. We know that $(L_1)_S = N_S$ and $(L_1)_C = N_C$ with $n_S = id_{L_{1S}}$ and $n_C = id_{L_{1C}}$. Furthermore g_3 is based on a source rule which means that $(G_{10})_T = (G_{20})_T$ and $g_{3_T} = id_{G_{10T}}$. Thus $\exists$ injective $q' : N \to G_{20}$ with $q' = (m'_{1S}, m'_{1C}, q_T)$ such that $q' \circ n = g_3 \circ m_1$. q' is a valid triple graph morphism by the same arguments as for Fig. 14. This means m_1' is not NAC consistent $\Rightarrow$ contradiction!

□

Proof of Theorem 1. This proof is based on the proof without NACs in [EEE+07] and the following triangle diagram.



In a first step we want to decompose the match consistent NAC-consistent TGT-sequence (1) with injective matches into an intermediate version

$$G_0 = G_{00} \xRightarrow{tr_{1S}} G_{10} \xRightarrow{tr_{1F}} G_{11} \xRightarrow{tr_{2S}} \ldots \xRightarrow{tr_{nS}} G_{n(n-1)} \xRightarrow{tr_{nF}} G_{nn} = G_n \tag{3}$$

which is match consistent and NAC-consistent.

In [EEE+07] it has been shown that any tr is equal to the E-concurrent rule $tr_S \star_E tr_F$ without NACs with $E = L_F$ - the left hand side of the forward rule. Using this result following Lemma 1 multiple times we are able to split the triple rules with NACs until we obtain sequence (3).

Thereafter we can reorder the rules until we have sequence (2). In [EEE+07] it has been shown that tr_{iS} and tr_{jF} are sequentially independent for $i > j$ without NACs. Following Lemma 2 multiple times finally leads to sequence (2) which is still match consistent and NAC-consistent.

Analogously we can transform sequence (2) back into sequence (1). The bijective correspondence follows from the bijective correspondence of the Concurrency Theorem and the Local Church-Rosser Theorem in conjunction with the equivalence of the NAC-consistency according to Lemma 1 and 2.

□

4 Completeness and Correctness of Model Transformations with NACs

Model transformations with NACs from models of the source language VL_{S0} to models of the target language VL_{T0} can be defined on the basis of forward rules as shown in [EEE+07] without NACs. Vice versa, it is also possible to define backward transformations from target to source graphs using derived backward rules leading to bidirectional model transformations. In this section we analyze completeness and correctness of model transformations. Main results are based on the composition and decomposition result in Thm. 1 in Sec. 3.

Definition 9 (Model Transformation) $MT = (G_S, G \overset{tr_F^*}{\Longrightarrow} H, H_T)$ is a model transformation from G_S to H_T, if $G \overset{tr_F^*}{\Longrightarrow} H$ is source consistent with NACs, where G_S and H_T are the source and target graphs of G and H, respectively.

As pointed out already source consistency with NACs of $G \overset{tr_F^*}{\Longrightarrow} H$ means that the forward sequence is controlled by the corresponding source sequence $\emptyset \overset{tr_S^*}{\Longrightarrow} G$ which generates G. Model transformations are correct and complete with respect to the source and target language $VL_S = proj_S(VL)$ and $VL_T = proj_T(VL)$ (see Def. 3).

Theorem 2 (Correctness with NACs) *Each model transformation* $MT = (G_S, G \overset{tr_F^*}{\Longrightarrow} H, H_T)$ *is correct, i.e.* $G_S \in VL_S$ *and* $H_T \in VL_T$.

Proof. $(G \overset{tr^*}{\Longrightarrow} H)$ source consistent $\Rightarrow \exists\, (\emptyset \overset{tr_S^*}{\Longrightarrow} G \overset{tr_F^*}{\Longrightarrow} H)$ match consistent and $G_S = H_S$ $\Rightarrow \exists\, (\emptyset \overset{tr^*}{\Longrightarrow} H)$ by Thm. 1 $\Rightarrow H \in VL$ and $H_T \in VL_T$ and $G_S = H_S \in VL_S$. □

Theorem 3 (Completeness with NACs) *For each* $H \in VL$: $\exists$ *model transformation* $MT = (G_S, G \overset{tr_F^*}{\Longrightarrow} H, H_T)$ *with* $G_S \in VL_S$, $H_T \in VL_T$. *This means in particular:*

- *For each* $H_T \in VL_T$: $\exists\, G_S \in VL_S$ *and model transformation* $MT = (G_S, G \overset{tr_F^*}{\Longrightarrow} H, H_T)$,
- *For each* $G_S \in VL_S$: $\exists\, H_T \in VL_T$ *and model transformation* $MT = (G_S, G \overset{tr_F^*}{\Longrightarrow} H, H_T)$.

Proof. $H \in VL \Rightarrow \exists\, (\emptyset \overset{tr^*}{\Longrightarrow} H) \overset{Thm.1}{\Longrightarrow} \exists$ match consistent $(\emptyset \overset{tr_S^*}{\Longrightarrow} G \overset{tr_F^*}{\Longrightarrow} H)$ and $G_S = H_S \Rightarrow$ $G_S \in VL_S, H_T \in VL_T$ and $G \overset{tr_F^*}{\Longrightarrow} H$ is source consistent $\Rightarrow MT = (G_S, G \overset{tr_F^*}{\Longrightarrow} H, H_T)$ is model transformation. □

Coming back to the example of a model transformation from class diagrams to database models, the relevance and value of the given theorems can be described from the more practical view.

Step	Forward Sequence Elements		Backward Sequence Elements	
	Matched	Created	Matched	Created
1	s1	c1,t1	t1	s1,c1
2	s1,c1,t1,s4,s9	c4	s1,c1,t1	s4,s9,c4
3	s1,s2,s7,c1,t1	c2,t2,t5	s1,c1,t1,t2,t5	s2,s7,c2
4	s1-s3,s6-s8,c1,t1,t2,t5	c3,t3,t6,t7	s1,c1,t1-t3,s2,c2,s7,t5-t7	c3,s3,s6,s8
5	s4,s5,s10,c4,t1,t3,t6	c5,t4,t8,t9	s4,c4,t1,t3,t4,t8,t9	c5,s5,s10

Table 1: Steps forward and backward model transformation

The resulting data base of the following model transformation is correctly typed and completely corresponds to the class diagram, which is the source model of the transformation.

Example 3 *Fig. 15 shows triple graph G_5 of the model transformation $(G_S = G_{0,S}, G_0 \xRightarrow{tr_F^*} G_5, G_T = G_{5,T})$ with the following forward sequence: $G_0 \xRightarrow{Class2Table} G_1 \xRightarrow{Subclass2Table} G_2 \xRightarrow{Attr2Col} G_3 \xRightarrow{NextAttr2NextCol} G_4 \xRightarrow{Attr2NextCol} G_5$, where G_0 is generated by the corresponding source sequence $\emptyset \xRightarrow{tr_S^*} G_0$. All elements are labeled with numbers specifying the matches and the created objects for each transformation step according to the left part of Table 1. G_S is given by G_5 restricted to elements of the class diagram part. After creating the table and building up the correspondences to the class nodes in the first two derivation steps, rules for translating attributes are applied. All steps of the sequence respect the NACs and furthermore, they correspond to a suitable source sequence making the forward transformation source consistent. In the third step, rule "Attr2Column" is applied and translates attribute "s2" to column "t2". Attribute s_3 is generated after s_2 in the source sequence, which is required by the source NAC of "NextAttr2NextColumn". Thus, the corresponding forward transformation translates s_3 after s_2. The remaining two attributes are translated by "NextAttr2NextColumn" and "Attr2NextColumn", where the target NACs ensure that the created columns are inserted after the last existing one of table "t1". Thus, the ordering of the created columns is not completely determined by the source model itself, but depends on the chosen source sequence. The nodes and edges of correspondence and target component as well as the morphisms $(G_{5,S} \leftarrow G_{5,C} \rightarrow G_{5,T})$ are created during the forward transformation.*

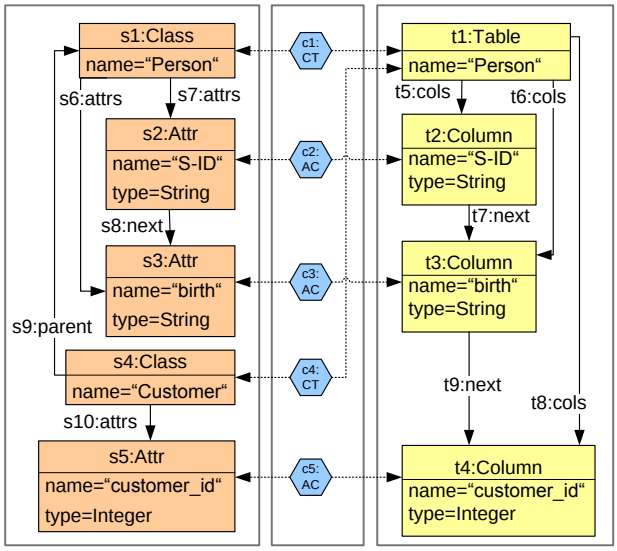


Figure 15: G_5 of Forward Sequence

5 Information Preserving Model Transformations

In [EEE+07] we have shown that there is an equivalence between corresponding forward and backward TGT sequences. This equivalence is based on the canonical decomposition and composition result, which is extended to the case with NACs in this paper (see Theorem 1).

Theorem 1 and its dual version lead to the following equivalence of forward and backward TGT-sequences with source-target NACs, which can be derived from the same general TGT-sequence.

Theorem 4 (Equivalence of Forward and Backward TGT-sequences with source-target NACs) *Each of the following TGT-sequences with source-target NACs implies the other ones where the matches are uniquely determined by each other.*

1. $G_0 \overset{tr_1}{\Longrightarrow} G_1 \overset{tr_2}{\Longrightarrow} G_2 \Longrightarrow \ldots \overset{tr_n}{\Longrightarrow} G_n$ (1)

2. $G_0 = G_{00} \overset{tr_{1S}}{\Longrightarrow} G_{10} \Longrightarrow \ldots \overset{tr_{nS}}{\Longrightarrow} G_{n0} \overset{tr_{1F}}{\Longrightarrow} G_{n1} \Longrightarrow \ldots \overset{tr_{nF}}{\Longrightarrow} G_{nn} = G_n$, (2)
which is match consistent. In this case we have: $G_{00,T} = G_{n0,T}$, $G_{n0,S} = G_{nn,S}$.

3. $G_0 = G_{00} \overset{tr_{1T}}{\Longrightarrow} G_{01} \Longrightarrow \ldots \overset{tr_{nT}}{\Longrightarrow} G_{0n} \overset{tr_{1B}}{\Longrightarrow} G_{1n} \Longrightarrow \ldots \overset{tr_{nB}}{\Longrightarrow} G_{nn} = G_n$, (3)
which is match consistent. In this case we have: $G_{00,S} = G_{0n,S}$, $G_{0n,T} = G_{nn,T}$.

Proof. Theorem 4 is a direct consequence of Theorem 1 concerning decomposition and composition of forward TGT-sequences with NACs and its dual version for target rules tri_T and backward rules tri_B where match consistency in Part 3 is defined by the T-components of the matches. □

Theorem 5 (Information Preserving Forward Transformation)
Each source consistent forward TGT-sequence $G \overset{tr_F^*}{\Longrightarrow} H$ *is backward information preserving, i.e. for* $K = (\emptyset \leftarrow \emptyset \rightarrow H_T)$, *there is a backward TGT-sequence* $K \overset{tr_B^*}{\Longrightarrow} H$, *which means that the source model* G_S *can be reconstructed from the target model* H_T*:*

$$G \overset{tr_F^*}{\Longrightarrow} H \overset{proj_T}{\longrightarrow} K \overset{tr_B^*}{\Longrightarrow} H \text{ with } G_S = H_S.$$

Proof. $G \overset{tr_F^*}{\Longrightarrow} H$ is source consistent which implies the existence of (2) $\emptyset \overset{tr_S^*}{\Longrightarrow} G \overset{tr_F^*}{\Longrightarrow} H$ being match consistent with $G_S = H_S$. By Theorem 4 with $G_0 = \emptyset$, $G_{n0} = G$, $G_{0n} = K$ and $G_n = H$ we obtain (3) $\emptyset \overset{tr_T^*}{\Longrightarrow} K \overset{tr_B^*}{\Longrightarrow} H$ being match consistent with $K_T = H_T$ and $H_S = G_S$ leading to $G \overset{tr_F^*}{\Longrightarrow} H \overset{proj_T}{\longrightarrow} K \overset{tr_B^*}{\Longrightarrow} H$. Hence, $G \overset{tr_F^*}{\Longrightarrow} H$ is backward information preserving. □

Example 4 *Example 3 Table 1 shows that for the given model transformation* $G_0 \overset{tr_F^*}{\Longrightarrow} G_5$ *according to Thm. 5 there is an inverse backward transformation* $G_5|_T \overset{tr_B^*}{\Longrightarrow} G_5$, *i.e. the source model can be reconstructed. However, there are also target consistent backward transformations* $G_5|_T \overset{tr_B^*}{\Longrightarrow} G'_5$ *with* $G'_{5,S} \neq G_{0,S}$, *because there are some class models with different inheritance relations corresponding to the given data base model.*

6 Related Work

Correctness of model transformations can be analyzed from different perspectives. Baleani et. al. motivate in [BFM+05] that correctness of model transformations for industrial tools should be based on formal models in order to ensure correctness by construction. For this purpose they suggest to use a block diagram formalism, called synchronous reactive model of computation (SR MoC). However, correct interpretation of the model transformation rules does not imply a correct result, such that it is a model of the target language. Semantical correctness is discussed by Karsai et. al. in [KN06], where specific behavior properties of the source model shall be reflected in the target model. This property can be checked for a restricted class of models. In [EE08] semantical correctness is ensured by using the rules for the model transformation also for the transformation of the operational semantics, which is given by graph rules. This way the behaviour of the source model can be compared with the one of the target model by checking mixed confluence. However, this paper concentrates on syntactical correctness based on the integrated language generated by the triple rules.

Our example in this paper presents a model transformation with NACs from class diagrams to relational data bases and it is based on the grammars defined in [EEE+07] and especially on [SK08]. In contrast to the presented algorithm in [SK08] for controlling the model transformations we introduced NAC consistency based on source consistent forward sequences. In this way we could extend several important results to the case of TGGs with NACs. In particular, model transformations given by source consistent forward transformations are correct and complete with respect to VL by Theorems 2 and 3. While a formal proof of correctness for the above mentioned algorithm is not given in [SK08], completeness of the algorithm is effectively not ensured, because recursion calls may cause transformations that produce structures forbidden by other necessary rule applications.

But still the algorithm in [SK08] convinces to be an elegant approach for a restricted class of relations to efficiently detect correct rule orderings for a subset of model transformations. This opens the possibility to combine efficiency with the here presented results in the following way: Each model transformation with NACs given by an efficient algorithm can be checked to be correct by performing the test of source consistency presented as Fact 2 in [EEH08c], which is now also valid for model transformations with NACs according to Thm. 1.

Model transformations based on triple rules with NACs were also analyzed in [EP08] for a restricted class of triple rules with distinct kernel elements. Special NACs of forward rules ensure that kernels are not translated twice and kernel typing guarantees that each rule produces exactly one kernel. For this restricted class of triple graph grammars local confluence and termination can be analyzed and thus, model transformations can be checked for functional behaviour.

7 Conclusion

This paper focusses on syntactical correctness and completeness. In order to analyze these important properties we extended the composition and decomposition result for triple graph transformations in [EEE+07] to the case with NACs, i.e. TGT sequences with NACs can be decomposed into source and forward as well as target and backward transformations, respectively, and vice

versa. Based on this fundamental property we have shown that source consistent model transformations are correct and complete with respect to the language given by the original triple rules. This extends the result in [EEH08a] to triple rules with NACs.

Source consistency of model transformations guarantees that each element of the source model was matched by a model transformation rule and correspondences to target model elements were created. A suitable source sequence can be calculated by parsing the source model using the source rules and the corresponding forward transformation can be checked to be source consistent. Alternatively, forward transformations can be created by an arbitrary strategy and checked afterwards using the algorithm for checking source consistency presented in [EEH08c]. Source consistency is not restricted to cases, where all source nodes have to be connected via correspondence nodes. Therefore, correctness of many algorithms for model transformations based on triple rules with NACs can be checked using the source consistency check.

According to [EEH08a] model integration sequences can be characterized as special model transformation sequences, such that the results of this paper for model transformation can be transferred to model integrations based on triple rules in a next step.

In this paper we focused on NACs which specify conditions on separately source and target elements. They are sufficient to most model transformations, which were considered by case studies so far. However, future work will include the analysis of how to handle general NACs and their relevance for language specification. An interesting problem - which could be solved with general NACs - is termination, where a parsing of the source model is omitted. A possibility may be to introduce additional NACs for the forward rules, such that source elements, which are already in correspondence with target elements, cannot be matched again for translation. In this way termination for a restricted class of rules could be ensured automatically. But note that NACs, which are equal to the right hand side of a forward rule, are not sufficient, because in this case matches of the transformation are required to be essential.

This paper has been published as technical report in [EHS09].

Bibliography

[BFM+05] M. Baleani, A. Ferrari, L. Mangeruca, A. L. Sangiovanni-Vincentelli, U. Freund, E. Schlenker, H.-J. Wolff. Correct-by-Construction Transformations across Design Environments for Model-Based Embedded Software Development. *Design, Automation and Test in Europe Conference and Exhibition* 2:1044–1049, 2005. doi:http://doi.ieeecomputersociety.org/10.1109/DATE.2005.105

[EE08] H. Ehrig, C. Ermel. Semantical Correctness and Completeness of Model Transformations using Graph and Rule Transformation. In *Proc. International Conference on Graph Transformation (ICGT'08)*. LNCS 5214, pp. 194–210. Springer Verlag, Heidelberg, 2008. http://tfs.cs.tu-berlin.de/publikationen/Papers08/EE08a.pdf

[EEE+07] H. Ehrig, K. Ehrig, C. Ermel, F. Hermann, G. Taentzer. Information Preserving Bidirectional Model Transformations. In Dwyer and Lopes (eds.), *Fundamental Ap-*

proaches to Software Engineering. LNCS 4422, pp. 72–86. Springer, 2007.
http://tfs.cs.tu-berlin.de/publikationen/Papers07/EEE+07.pdf

[EEH08a] H. Ehrig, K. Ehrig, F. Hermann. From Model Transformation to Model Integration based on the Algebraic Approach to Triple Graph Grammars. In Ermel et al. (eds.), *Proc. Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT'08)*. Volume 10. EC-EASST, 2008.
http://eceasst.cs.tu-berlin.de/index.php/eceasst/issue/view/19

[EEH08b] H. Ehrig, C. Ermel, F. Hermann. On the Relationship of Model Transformations Based on Triple and Plain Graph Grammars. In Karsai and Taentzer (eds.), *Proc. Third International Workshop on Graph and Model Transformation (GraMoT'08)*. ACM, New York, NY, USA, 2008.
doi:http://doi.acm.org/10.1145/1370175.1370244
http://tfs.cs.tu-berlin.de/publikationen/Papers08/EEH08.pdf

[EEH08c] H. Ehrig, C. Ermel, F. Hermann. On the Relationship of Model Transformations Based on Triple and Plain Graph Grammars (Long Version). Technical report 2008/05, Technische Universität Berlin,Fakultät IV, 2008.
http://iv.tu-berlin.de/TechnBerichte/2008/2008-05.pdf

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theoretical Computer Science. Springer Verlag, 2006.
http://www.springer.com/3-540-31187-4

[EHS09] H. Ehrig, F. Hermann, C. Sartorius. Completeness and Correctness of Model Transformations based on Triple Graph Grammars with Negative Application Conditions (Long Version). Technical report 2009/3, TU Berlin, 2009.
http://www.eecs.tu-berlin.de/fileadmin/f4/TechReports/2009/tr-2009-03.pdf

[EP08] H. Ehrig, U. Prange. Formal Analysis of Model Transformations Based on Triple Graph Rules with Kernels. In Ehrig et al. (eds.), *Proc. International Conference on Graph Transformation (ICGT'08)*. LNCS 5214, pp. 178–193. Springer Verlag, Heidelberg, 2008.
http://tfs.cs.tu-berlin.de/publikationen/Papers08/EP08a.pdf

[HHT96] A. Habel, R. Heckel, G. Taentzer. Graph Grammars with Negative Application Conditions. *Special issue of Fundamenta Informaticae* 26(3,4):287–313, 1996.

[KN06] G. Karsai, A. Narayanan. On the Correctness of Model Transformations in the Development of Embedded Systems. In Kordon and Sokolsky (eds.), *Monterey Workshop*. LNCS 4888, pp. 1–18. Springer, 2006.

[KS06] A. Königs, A. Schürr. Tool Integration with Triple Graph Grammars - A Survey. In *Proc. SegraVis School on Foundations of Visual Modelling Techniques*. Volume 148, pp. 113–150. Electronic Notes in Theoretical Computer Science, Elsevier Science,

Amsterdam, 2006.
http://www.sciencedirect.com/science/journal/15710661

[Sch94] A. Schürr. Specification of Graph Translators with Triple Graph Grammars. In Tinhofer (ed.), *WG94 20th Int. Workshop on Graph-Theoretic Concepts in Computer Science*. LNCS 903, pp. 151–163. Springer Verlag, Heidelberg, 1994.

[SK08] A. Schürr, F. Klar. 15 Years of Triple Graph Grammars. In Ehrig et al. (eds.), *ICGT*. LNCS 5214, pp. 411–425. Springer, 2008.
doi:10.1007/978-3-540-87405-8_28